Specifikacija i modeliranje softvera

Dijagram klasa

Aleksandra Mitrović
amitrovic@singidunum.ac.rs

Univerzitet Singidunum
Centar Novi Sad

20. april 2021.

Sadržaj

Sadržaj

Zašto dijagram klasa?

- Dijagramom klasa možemo specificirati skicu ili kompletno rešenje, počevši od ranih faza razvoja softverskog sistema.
- U ranim fazama je često površan i uglavnom služi za komunikaciju ideja u okviru projektnog tima.
- U kasnijim fazama je detaljniji i služi za specifikaciju implementacije, generisanje koda i dokumentovanje rešenja.

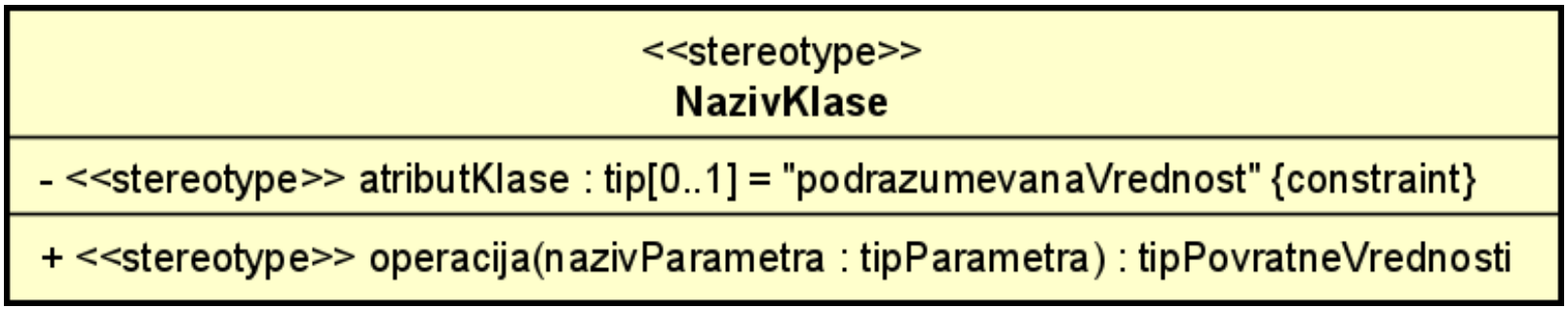
Sadržaj

Elementi dijagrama

- Osnovni elementi dijagrama klasa su:
  - klasa,
  - interfejs,
  - paket.

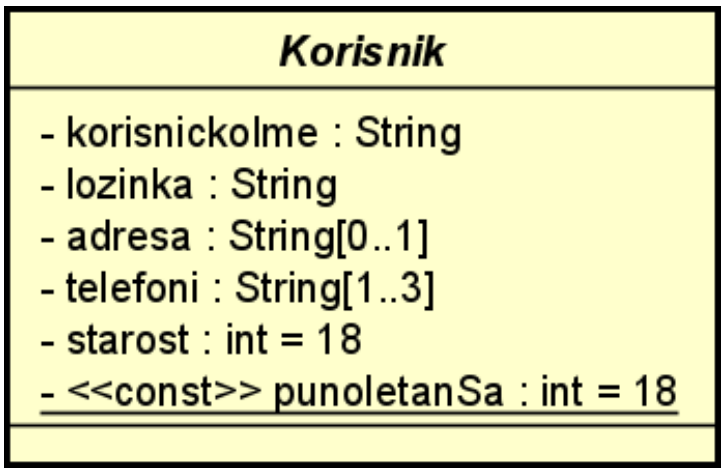
Klasa

- Klasama definišemo tipove objekata.
- Nazivi klasa su obično imenice iz specifikacije (npr. Automobil, Osoba, ...).
- Klasu opisujemo atributima i metodama. Dodatno možemo specificirati modifikatore i stereotipe.



Atributi klase

- Atributi ili obeležja dodatno opisuju klasu.
- Osnovne osobine atributa su vidljivost, naziv, tip podatka, podrazumevana vrednost i kardinalitet.
- Dodatno mogu imati stereotipe i ograničenja.



Vidljivost

- Vidljivost obeležja i operacija nam označava pravo pristupa istim i može biti:

Simbol	Naziv	Opis
-	private	Privatno - pristup je dozvoljen samo u okviru klase. Naslednici ne mogu direktno pristupati.
+	public	Javno - pristup je moguć bez ograničenja.
#	protected	Zaštićeno - pristup u okviru klase i njihovih naslednica.
~	package	Paketno - pristup omogućen svim klasama u okviru paketa.

*Mapiranje modifikatora pristupa ne mora se nužno poklopiti sa vidljivošću atributa u programskom kodu.

Tipovi podataka

- Tipovi podataka mogu biti:

Naziv	Opis
predefinisan	Osnovni tipovi podataka poput `int`, `real`, `float`, `boolean`, `byte`, `string`, `char` i drugi.
nabrojivi	Tip definisan preko enumeracije.
klasa	Korisnički definisana ili klasa definisana u nekom konkretnom jeziku ili biblioteci.
interfejs	Korisnički definisan ili interfejs definisan u nekom konkretnom jeziku ili biblioteci.
*Mapiranje predefinisanih tipova ne mora biti identično u programskom kodu.	

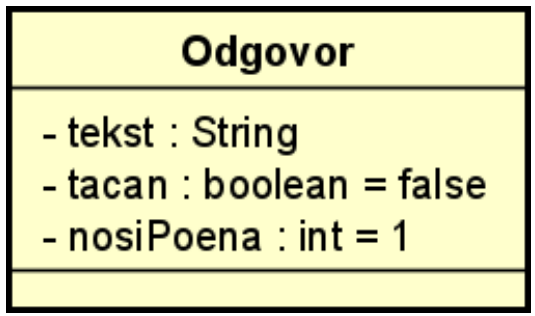
Kardinalitet

- Kardinalitet nam omogućava da definišemo koliko pojava tipa podatka se nalazi u obeležju. Kardinaliteti mogu biti:

Naziv	Opis
`1`	Tačno jedna pojava.
`0..1`	Nula ili jedna pojava (`nullable`).
`0..*`	Nula ili više (predstavlja kolekciju).
`1..*`	Jedan ili više (predstavlja kolekciju).
`m..n`	Interval `[m, n]` (`n` je veće ili jednako `m`).
`*`	Nula, jedan ili više (predstavlja kolekciju).
*Ako se izostavi, podrazumevano je 1 pojava.	

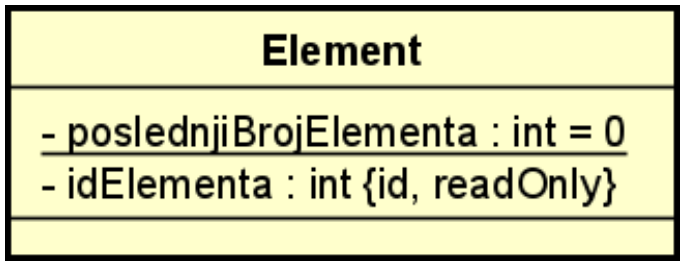
Podrazumevana vrednost

- Podrazumevana vrednost je vrednost koju obeležje dobija prilikom instanciranja klase.



Statičko obeležje

- Statičko obeležje je ono kojem se pristupa na nivou klase, a ne na nivou instance.
- Koristimo ga za skladištenje vrednosti koja je potrebna svim instancama klase, za implementaciju globalno dostupnih resursa, za specifikaciju konstanti.



Ograničenja

- Ograničenja dodatno naglašavaju razvojnom timu ili generatoru koda kako programski kod treba da se implementira, mogu biti:

Naziv	Opis
`id`	Jedinstveni identifikator klase. Pri mapiranju na obeležje u bazi podataka označava deo primarnog ključa.
`readOnly`	Zabranjena izmena nakon inicijalizacije.
`derived`	Izvedeno obeležje na osnovu drugih obeležja.
`ordered`	Označava da je obeležje uređena kolekcija elemenata.
`unique`	Označava da u kolekciji ne sme biti ponavljanja elemenata sa istom vrednošću.
`nonunique`	Označava da u kolekciji može biti ponavljanja elemenata sa istom vrednošću.

Operacije klase

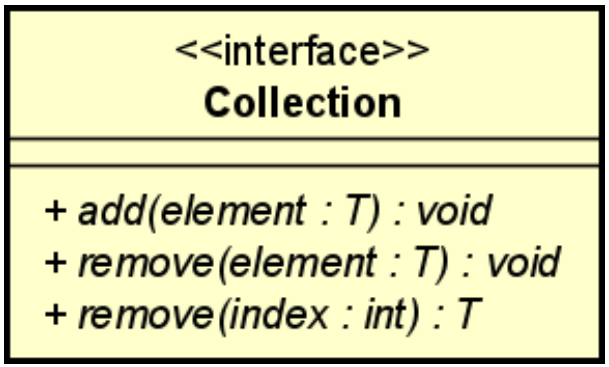
- Operacije opisuju funkcionalnosti klase i kroz njih definišu ponašanje. U dijagramu klasa se ne opisuje ponašanje, već se specificira samo potpis operacije.
- U programskim jezicima odgovoraju funkcijama i metodama klase.
- U specifikaciji operacije navodimo vidljivost, naziv, listu parametara, tip povratne vrednosti. Dodatno možemo specificirati i modifikatore.

Operacije klase

- Vidljivost operacije se specificira isto kao i kod obeležja.
- Tip povratne vrednosti se specificira isto kao i kod obeležja.
- Operacije takođe mogu biti statičke i apstraktne.

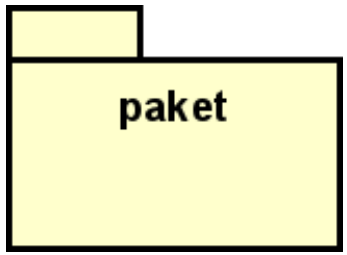
Interfejs

- Interfejsima definišemo nove tipove i šablone za proširivanje u drugim klasama.
- Klase koje implementiraju interfejse je potrebno da na svoj način definišu potpise metoda specificiranih u interfejsu.



Paket

- Pakete koristimo za grupisanje srodnih klasa i interfejsa.
- Paketom definišemo prostor imena.



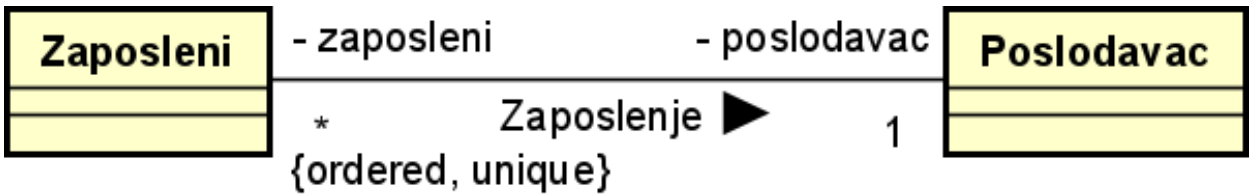
Sadržaj

Relacije

- Elemente dijagrama klasa povezujemo vezama:
  - asocijacije,
  - generalizacije,
  - realizacije i
  - zavisnosti.

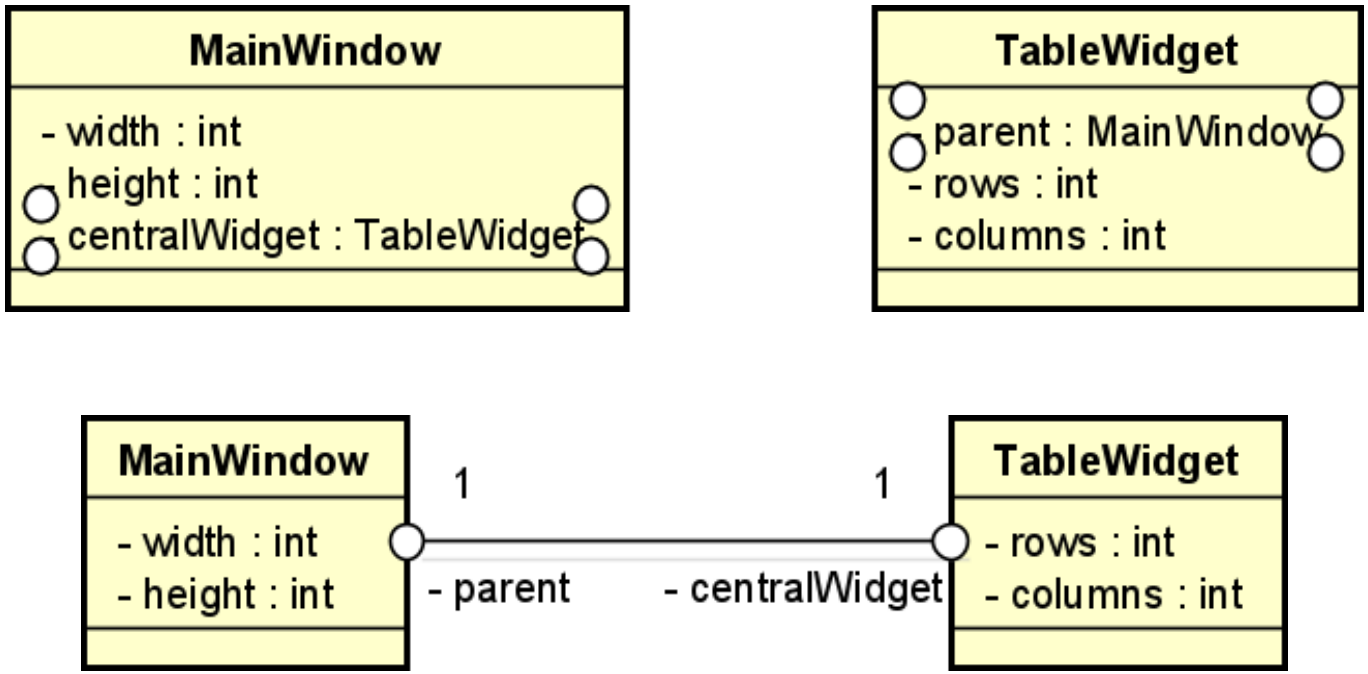
Veza asocijacije

- Omogućava uvezivanje klasa pri kojem se stiču obeležja tipa klase sa kojom je uvezana.
- Asocijacija je opisana nazivom, kardinalitetima i nazivima obeležja koji se formiraju na njenim krajevima.
- Vezu asocijacije prepoznajemo kao glagol u specifikaciji sistema koji se modeluje. Na primer: Zaposleni *radi* kod poslodavca.



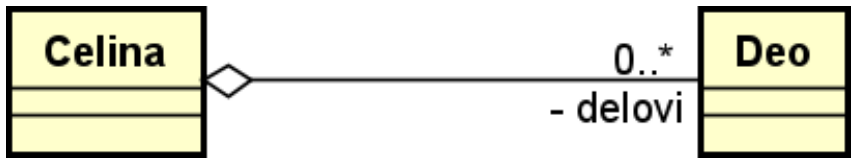
Veza asocijacije

- Prepoznavanje asocijacije u modelu.



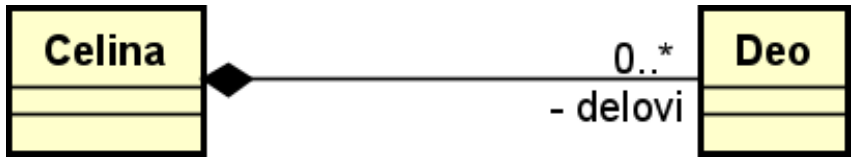
Veza agregacije

- Agregacija predstavlja specijalizaciju asocijacije, čime dodaje značenje celine i dela. Celina se formira grupisanjem delova, ali delovi mogu postojati i bez celine.
- Vezu agregacije prepoznajemo kao glagol *sastoji se od* u specifikaciji sistema.



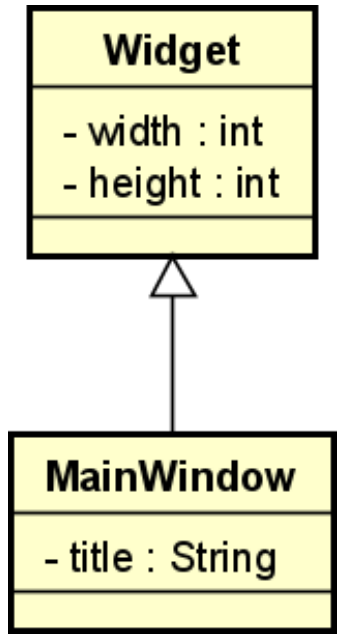
Veza kompozicije

- Kompozicija predstavlja specijalizaciju agregacije, pri čemu je dodato ograničenje da deo ne može postojati bez celine.
- U vezama agregacije i kompozicije kardinalitet kraj celine je uvek 1.



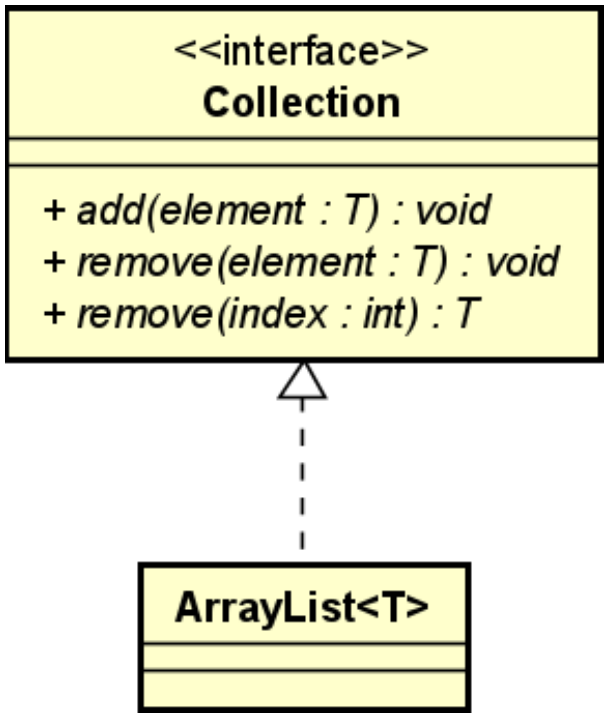
Veza generalizacije

- Generalizacija povezuje opšte elemente modela sa specijalizovanim. Može se koristiti između klasa koje su iste vrste i dele zajedničke osobine i ponašanje.
- Nasleđuju se sva obeležja i operacije opšte klase, pri čemu se ona mogu proširiti novim.
- Vezu generalizacije prpoznajemo u specifikaciji sistema preko izraza *su*, *mogu biti*. Primer: Klijenti banke *mogu biti* fizička i pravna lica.



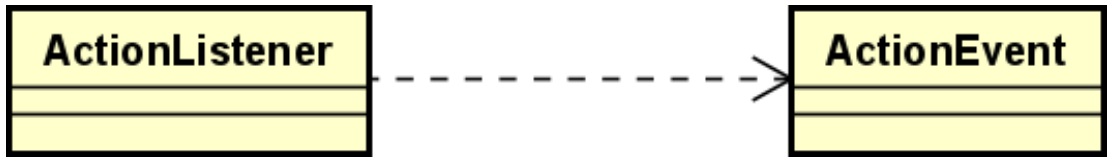
Veza realizacije

- Realizacijom interfejsa klasa dodatno opisuje metode potpisane u interfejsu.
- Slabija je veza od generalizacije, jer klase koje implementiraju (realizuju) interfejs, ne moraju imati ništa zajedničko od atributa.



Veza zavisnosti

- Veza zavisnosti označava da jedna klasa zavisi od druge, ali ne na način kao što su prethodne veze označavale, iako i one predstavljaju zavisnost.
- Uglavnom se koristi kada želimo da naglasimo da jedna klasa kreira neku drugu klasu, koristi njene usluge, ili prenosi instancu klase kao argument u svojim metodama, bez toga da je čuva u svojim atributima.



Sadržaj

Literatura

- Knjige:
  - Martin Fowler - UML Distilled.
  - Gordana Milosavljević - Uvod u modelovanje softvera.
- Veb stranice:
  - https://www.uml.org/
  - https://www.uml-diagrams.org/